IN THE SPECIFICATION

Amendments to the Specification:

Please insert the following paragraph immediately after line 10 on page 9 of the specification as originally filed, which corresponds to immediately after paragraph 30 of the application as published as U.S. Patent Application Publication 20020095500:

--Figure 13 is a block diagram showing the re-partitioning of functionality between the operating system and the compute capsule.--

Please insert the following 16 paragraphs, including the subheading "Re-Partitioning the Operating System" immediately before the sub-heading "Resource Aggregation Units" on page 11 of the specification as originally filed, or immediately after paragraph 34 of the application as published as U.S. Patent Application Publication 20020095500:

--Re-Partitioning the Operating System

To provide such functionality, the traditional operating system is re-partitioned as shown in FIG. 13 so that all host-dependant and personalized elements of the computing environment are moved into the capsule 1300, while leveraging policies and management of the shared underlying system 1305. The computing environment comprises CPU 1310, file system 1315, devices 1320, virtual memory 1325, and IPC 1330. Each of these components of the computing environment have been partitioned as indicated by the curved line 1335.

The state of the CPU scheduler 1340 is left in the operating system 1305. This state comprises information that the operating system maintains so that it knows which processes may run, where they are, what priority they have, how much time they will be granted processor attention, etc. Process state 1345, which is moved to the compute capsule 1300, has process-specific information, such as the values in the registers, the signal handlers registered, parent/child relationships, access rights, and file tables. The file system 1315 leaves local files 1350 that are identically available on all machines, (e.g., /usr/bin or /man on a UNIX system) in the operating system 1305. The file system 1315 further leaves disk blocks 1352

outside the capsule, which are caches of disk blocks that are read into the system and can be later used when needed to be read again. The disk structure 1354 is also left outside the capsule. The disk structure is specific to an operating system and serves as a cache of where files are located on the disk, (i.e., a mapping of pathnames to file locations). Network file system (NFS) is a protocol for accessing files on remote systems. The operating system maintains information 1356 with respect to the NFS and a cache 1358, which is a cache of files the operating system has retrieved from remote servers and stored locally. Similar state is maintained for other network based file systems.

What has been partitioned away from the operating system is the file state 1360. The file state 1360 is moved to the capsule 1300. The file state 1360 is the state of a file that some process in the capsule has opened. File state 1360 includes, for instance, the name of the file and where the process currently accessing the file. If the file is not accessible via the network (e.g., stored on a local disk), then its contents are placed in the capsule.

Devices 1320 are components that are attached to the computer. For each device there is a driver that maintains the state of the device. The disk state 1365 remains in the operating system 1305. The other device components are specific to a log-in session and are moved to the capsule 1300. The other devices include a graphics controller state 1370, which is the content that is being displayed on the screen, for instance the contents of a frame buffer that holds color values for each pixel on a display device, such as a monitor.

Keyboard state 1372 and mouse state 1375 includes the state associated with the user's current interaction with the keyboard, for instance whether caps lock is on or off and with the screen, for instance where the pointer is currently located. Tty state 1374 includes information associated with the terminals the user is accessing, for instance if a user opens an Xwindow on a UNIX system or if a user telnets or performs an rlogin. Tty state 1374 also includes information about what the cursor looks like, what types of fonts are displayed in the terminals, and what filters should be applied to make the text appear a certain way, for instance.

Virtual memory 1325 has state associated with it. The capsule tracks the state associated with changes made from within the capsule which are termed read/write pages 1376. Read-only pages 1378 remain outside the capsule. However, in one embodiment read-only pages 1378 are moved to the capsule as well, which is useful in some scenarios. For instance, certain commands one would expect to find on a new machine when their capsule

migrates there may not be available. Take, for instance, a command such as Is or more on a UNIX system. Those read-only pages may not be necessary to bring into the capsule when it is migrating between UNIX machines, because those pages exist on every UNIX machine. If, however, a user is moving to a machine that does not use those commands, it is useful to move those read only pages into the capsule as well. The swap table 1380, which records what virtual memory pages have been replaced and moved to disk, remains outside the capsule as do the free list 1382, (which is a list of empty virtual memory pages), and the page table 1384.

Nearly all IPC 1330 is moved into the capsule. This includes shared memory 1386, which comprises a portion of memory that multiple processes may be using, pipes 1388, fifos 1390, signals 1392, including handler lists and the state needed to know what handler the process was using and to find the handler. Virtual interface and access control 1394 is useful for separating the capsule from host-dependent information that is specific to a machine, such as the structure of internal program state or the IDs for its resources. The interface 1394 refers generally to the virtualized naming of resources and translations between virtual resource names and physical resources, as well as lists that control access to processes trying to access capsules.

Thus, capsule state includes data that are host-specific, cached on the local machine to which the capsule is bound, or not otherwise globally accessible. This includes the following information:

Capsule State: Name translation tables, access control list, owner ID, capsule name, etc.;

Processes: Tree structure, process control block, machine context, thread contexts, scheduling parameters, etc.;

Address Space Contents: Read/write pages of virtual memory; because they are available in the file system, contents of read-only files mapped into the address space (e.g., the application binary and libraries) are not included unless explicitly requested;

Open File State: Only file names, permissions, offsets, etc. are required for objects available in the global file system. However, the contents of personal files in local storage (e.g., /tmp) must be included. Because the pathname of a file is discarded after it is opened, for each process one embodiment of the invention maintains a hash table that maps file descriptors to their corresponding pathnames. In addition, some open files have no pathname,

(i.e., if an unlink operation has been performed). The contents of such files are included in the capsule as well;

IPC Channels: IPC state has been problematic in most prior systems. The present invention adds a new interface to the kernel modules for each form of IPC. This interface includes two complementary elements: export current state, and import state to re-create channel. For example, the pipe/fifo module is modified to export the list of processes attached to a pipe, its current mode, the list of filter modules it employs, file system mount points, and in-flight data. When given this state data, the system can re-establish an identical pipe.

Open Devices: By adding a state import/export interface similar to that used for IPC, the invention supports the most commonly used devices: keyboard, mouse, graphics controller, and pseudo-terminals. The mouse and keyboard have very little state, mostly the location of the cursor and the state of the LEDs (e.g., caps lock). The graphics controller is more complex. The video mode (e.g., resolution and refresh rate) and the contents of the frame buffer must be recorded, along with any color tables or other specialized hardware settings. Supporting migration between machines with different graphics controllers is troublesome, but a standard remote display interface can address that issue. Pseudo-terminal state includes the controlling process, control settings, a list of streams modules that have been pushed onto it, and any unprocessed data.

Capsules do not include shared resources or the state necessary to manage them (e.g., the processor scheduler, page tables), state for kernel optimizations (e.g., disk caches), local file system, physical resources (e.g., the network), etc.--